CLAIMS

WE CLAIM:

1. A method for providing a topology interface for a multimedia processing system, the method comprising:

receiving a plurality of media parameters identifying at least an identifier, a node type, a data type and a duration; and

in response, creating a topology capable of being passed to a media processor as an extensible symbolic representation of an intended media flow.

2. The method of claim 1 wherein the media parameters include one or more of a GetCacherObject, a GetNodeType, a GetTopoNodeID, a SetProjectStartStop, a GetProjectStartStop, a GetInputCount, a GetOutputCount, a ConnectOut, a GetInput, a GetOutput, a SetOutputPrefType, a GetOutputPrefType, a SetMajorType, a GetMajorType, a CloneFrom, a SetInputCount, a SetOutputCount, a SetStreamDiscardable, a GetStreamDiscardable, a SetOptionalFlag, and a GetOptonalFlag.

3. The method of claim 1 wherein the media parameters include a SetSourceAndDescriptor method that enables a topology loader to create a media stream based on a descriptor.

4. The method of claim 1 wherein the node type is a segment topology node type such that any modifications made to the topology to add, remove or connect nodes does not alter input and output nodes.

5. The method of claim 1 wherein the unique identifier enables sharing and reusing the nodes in a plurality of topologies.

6. The method of claim 4 wherein the segment topology node type is created via an IMFSegmentTopologyNode : IUnknown interface.

7. The method of claim 4 wherein the segment topology node type is created via an IMFSegmentTopologyNode : IUnknown interface including one or more of GetSegmentTopology(IMFTopolgy* pTopology), SegmentTopology(IMFTopology** ppTopology), SetDirty(BOOL bDirty), BOOL IsDirty(), BOOL GetActualOutputNode(long lOutputIndex, IMFTopologyNode** ppActualNode, long* plNodeOutputIndex), and BOOL GetActualInputNode(long lInputIndex, IMFTopologyNode** ppActualNode, long* plNodeInputIndex).

8. A software architecture for providing a topology of media objects for a media processing system, the software architecture comprising:

at least a first component that identifies a connection between one or more nodes in the media processing system;

at least a second component that abstracts the connection between the nodes to enable the topology to be fully or partially specified independent of instantiation of the media objects; and

at least one application program interface to access the first and second components.

9. A method for providing an interface for a media processing system to provide a split topology node, the method comprising:

receiving a media type parameter;

receiving a descriptor parameter, the descriptor parameter configured to be updated and streams are selected/deselected based on one or more output nodes connected to the outputs of the split topology node.

10. The method of claim 9 wherein the interface operates on one or more commands including SetInputMediaType(IMFMediaType* pPartialMediaType), GetInputMediaType(IMFMediaType** ppPartialMediaType), InitOutputDescriptor(IMFPresentationDescriptor* pPresentationDescriptor), and GetOutputDescriptor(IMFPresentationDescriptor** ppPresentationDescriptor.

11. A computer readable medium on which is stored a topology function comprising:

a first input parameter representing a unique identifier;

a second input parameter representing a state of a topology;

a third parameter representing a descriptor for the topology;

a fourth parameter representing one or more characteristics about a node of the topology; and

executable instructions adapted to provide a topology capable of being passed to a media processor as an extensible symbolic representation of an intended media flow calculated based on at least one of the input parameters.

12. The computer readable medium of claim 11 wherein the unique identifier is input via a GetTopologyID(TOPOID * pID) command.

13. The computer readable medium of claim 11 wherein the state of the topology is input via a GetState(MF_TOPOLOGY_STATE *pState) command.

14. The computer readable medium of claim 11 wherein the state of the topology is one or more of a partial, loading, loaded, and full topology.

15. The computer readable medium of 14 wherein:

a partial topology state includes a state with a specified general flow of bits and one or more source and object is independent of automatic interfacing upon loading;

a loading topology state includes a state for which a topology loader is actively engaged in altering a topology from a partial state or full state to a loaded state;

a loaded topology state includes a state for which each source or object in the topology has already been loaded and contains a pointer to the object independent of agreement on one or more media types; and

a full topology state includes a state for which each source or object has an agreed upon media type and the topology is ready for processing.

16. The computer readable medium of claim 13 wherein the state of the topology is identified via a 32 bit hexadecimal value.

17. The computer readable medium of claim 11 wherein the descriptor is input via a GetTopologyDescriptor(IMFTopology** ppTopologyDescriptor) command.

18. The computer readable medium of claim 11 wherein the one or more characteristics about a node of the topology are input via one or more of a AddNode(IMFTopologyNode *pNode), RemoveNode(IMFTopologyNode * pNode), GetNodeCount(WORD *pcNodes), GetNode(WORD wIndex, IMFTopologyNode **ppNode), Clear( ) and CloneFrom(IMFTopology* pTopology) command.

19. A method for providing a segment topology node interface for a multimedia processing system, the method comprising:

receiving a first parameter defining one or more connections for the segment topology node;

receiving a second parameter identifying a pointer to a topology to which the segment topology node can connect; and

in response, creating the segment topology node as part of a topology that is incapable of alteration of input and output nodes to the segment topology node, the segment topology node being separately identifiable.

20. The method of claim 19 wherein the segement topology node is created by a topology loader operable through one or more of a SetSegmentTopology(IMFTopolgy* pTopology) command, a GetSegmentTopology(IMFTopology** ppTopology) command, a SetDirty(BOOL bDirty) command, a IsDirty() command, a GetActualOutputNode(long lOutputIndex, IMFTopologyNode** ppActualNode, long* plNodeOutputIndex) command and a GetActualInputNode(long lInputIndex, IMFTopologyNode** ppActualNode, long* plNodeInputIndex) command.

21. The method of claim 20 wherein the IsDirty and the SetDirty commands relate to a dirty flag on the topology that is inside the segment topology node to determine whether the topology requires resolving.

22. The method of claim 20 wherein the GetActualOutputNode command and the GetActualInputNode command are used to find a base level non-segment node connected to one of an output stream and an input stream at a predetermined index of the segment topology node.

23. A method for providing an interface for a multimedia processing system, the method comprising:

receiving a media processor parameter related to received media data;

receiving a timeline parameter related to timing of events to occur for performing media processing; and

receiving a topology parameter describing a flow for the received media data; and

in response, enabling a multimedia processing function via an extensible symbolic abstraction of media objects related to one or more of the media processor parameter, the timeline parameter and the topology parameter.

24. A method for identifying a flow of multimedia data through a collection of one or more media objects forming one or more nodes, the method comprising:

providing at least a first component that identifies a connection between one or more nodes;

providing at least a second component that abstracts the connection between the nodes to enable a topology to be fully or partially specified independent of instantiation of the media objects; and

providing an application programming interface to access at least one of the first and second components.

25. The method of claim 24 wherein the abstracting the connection between the nodes enables a delay between negotiating one or more media types for the topology and loading the media objects.

26. The method of claim 24 wherein the topology includes a segment topology node configured to provide an encapsulated topology that can be inserted and deleted from the topology, the segment topology node including one or more inputs and one or more outputs.

27. The method of claim 24 wherein the topology includes a tee node configured to provide a primary and secondary output stream therefrom, the tee node configured to

respond to logic dictating a discardability of data output from one or more of the primary and the secondary output stream.

28. The method of claim 24 wherein the topology includes a demultiplexer node configured to split media into different types of media from a combined input.

29. The method of claim 28 wherein the combined input is an interleaved audio and video input, the demultiplexer node configured to split the audio from the video and provide at least an audio output and a video output.

30. The method of claim 24 wherein each node is identifiable via a unique identifier.

31. The method of claim 24 wherein the topology is identified by one or more topology descriptors enabling interaction between a user and the topology.

32. The method of claim 31 wherein the topology descriptor identifies a collection of topology stream descriptors, each topology stream descriptor identifying a media stream.